Exploring Deep Reinforcement Learning-Assisted Federated Learning for Online Resource Allocation in Privacy-Preserving EdgeIoT

Zheng, Jingjing ; Li, Kai; Mhaisen, Naram; Ni, Wei; Tovar, Eduardo ; Guizani, Mohsen

DOI
10.1109/JIOT.2022.3176739

Publication date
2022

Document Version
Final published version

Published in
IEEE Internet of Things Journal

Exploring Deep-Reinforcement-Learning-Assisted Federated Learning for Online Resource Allocation in Privacy-Preserving EdgeIoT

Jingjing Zheng, Kai Li, *Senior Member, IEEE*, Naram Mhaisen, Wei Ni, *Senior Member, IEEE*, Eduardo Tovar, *Member, IEEE*, and Mohsen Guizani, *Fellow, IEEE*

***Abstract*—Federated learning (FL) has been increasingly considered to preserve data training privacy from eavesdropping attacks in mobile-edge computing-based Internet of Things (EdgeIoT). On the one hand, the learning accuracy of FL can be improved by selecting the IoT devices with large data sets for training, which gives rise to a higher energy consumption. On the other hand, the energy consumption can be reduced by selecting the IoT devices with small data sets for FL, resulting in a falling learning accuracy. In this article, we formulate a new resource allocation problem for privacy-preserving EdgeIoT to balance the learning accuracy of FL and the energy consumption of the IoT device. We propose a new FL-enabled twin-delayed deep deterministic policy gradient (FL-DLT3) framework to achieve the optimal accuracy and energy balance in a continuous domain. Furthermore, long short-term memory (LSTM) is leveraged in FL-DLT3 to predict the time-varying network state while FL-DLT3 is trained to select the IoT devices and allocate the transmit power. Numerical results demonstrate that the proposed FL-DLT3 achieves fast convergence (less than 100 iterations) while the FL accuracy-to-energy consumption ratio is improved by 51.8% compared to the existing state-of-the-art benchmark.**

***Index Terms*—Deep reinforcement learning (DRL), federated learning (FL), Internet of Things, mobile-edge computing (MEC), online resource allocation.**

I. Introduction

MOBILE-EDGE computing (MEC) provides a promising solution to enabling cloud computing services in privacy-persevering MEC-based Internet of Things (EdgeIoT) [1]–[3]. The IoT devices can offload their local computation-intensive tasks to computationally powerful edge servers [4], [5]. In EdgeIoT, offloading the source data of the IoT devices to the edge server is vulnerable to eavesdropping attacks [6], [7]. To prevent private data leakage of the IoT devices, federated learning (FL) [8] is used to train a global data learning model at the edge server by aggregating the data structure parameters of the IoT devices, while the source data remains at the IoT devices.

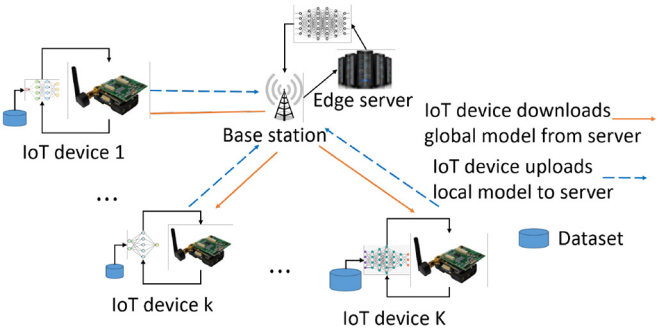


Fig. 1. FL-enabled EdgeIoT. A local model is trained with the data set at the IoT device. The local model is aggregated by an edge server, where the global model is trained and returned to the IoT devices.

Fig. 1 depicts an FL-enabled EdgeIoT, where the IoT devices are deployed to sense and process private information, e.g., health reports of patients [9]. Specifically, mobile phones use image classification models to classify pictures or images. FL can help multiple mobile phones to cooperatively train an effective global image classification model, without the need of sharing the images and pictures used for the training [10]. Likewise, vehicles can train an accurate autonomous driving model by aggregating the local models trained separately by multiple vehicles [11].

A local model (e.g., the weight vector [12]–[14] or gradient [15]) is trained on the sensing data of an IoT device to sense and process private information, such as private health reports of patients. Next, the edge server aggregates the local models of all IoT devices to create a comprehensive and effective global model without collecting the private data of the IoT devices. A global model is obtained at the edge server, e.g., by applying federated averaging (FedAvg) [12] to the local models. The edge server broadcasts the global model back to all the IoT devices. According to the global model, each of the IoT devices renews the training of its local model. By

Manuscript received 11 February 2022; revised 24 April 2022; accepted 13 May 2022. Date of publication 20 May 2022; date of current version 24 October 2022. This work was supported in part by the National Funds through FCT/MCTES (Portuguese Foundation for Science and Technology), within the CISTER Research Unit under Grant UIDP/UIDB/04234/2020, and in part by the National Funds through FCT under Project PTDC/EEI-COM/3362/2021 (ADANET). *(Corresponding author: Kai Li.)*

Jingjing Zheng, Kai Li, and Eduardo Tovar are with the Real-Time and Embedded Computing Systems Research Center, 4249-015 Porto, Portugal (e-mail: zheng@isep.ipp.pt; kai@isep.ipp.pt; emt@isep.ipp.pt).

Naram Mhaisen is with the College of Electrical Engineering, Mathematics, and Computer Science, TU Delft, 2628 CD Delft, The Netherlands (e-mail: n.mhaisen@tudelft.nl).

Wei Ni is with Commonwealth Scientific and Industrial Research Organization, Sydney, NSW 2122, Australia (e-mail: wei.ni@data61.csiro.au).

Mohsen Guizani is with the Machine Learning Department, Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE (e-mail: mguizani@ieee.org).

Digital Object Identifier 10.1109/JIOT.2022.3176739

iteratively training the local model at the IoT device and updating the global model at the edge server, the learning accuracy of FL on the data classification and event prediction can be progressively improved [16].

While selecting the IoT devices with large training data sets can improve the learning accuracy of FL [17], it can often result in fast depletion of the batteries at the devices. On the other hand, selecting the IoT devices with small data for training the local models can save the battery energy of the IoT devices, but likely leads to a low accuracy of the global model. In this sense, balancing the learning accuracy of FL and the energy consumption of the IoT devices is crucial to EdgeIoT.

Moreover, the FL process is time slotted by design and sequential decision making is required. Hence, in this article, we propose an online resource allocation optimization to balance the learning accuracy of FL and energy consumption of the IoT devices. In practice, the instantaneous information of data size, transmit power, and link qualities between the edge server and the IoT devices is unlikely to be known. The optimization is formulated as a partially observable Markov decision process (POMDP), where the network state consists of the source data size of the IoT devices, channel conditions between the edge server and the IoT devices, bandwidth, and the remaining energy of the selected IoT devices. All the entries of a network state are continuous. The action space includes the discrete selection of IoT devices and the continuous transmit power allocation. Given the continuous network states and actions, a large number of IoT nodes lead to a large state and action space, and complex network state transitions. This results in difficulty in the joint optimization of IoT device selection and transmits power allocation. Due to a large and continuous state and action space in the formulated POMDP, a new deep reinforcement learning (DRL)-based device selection and transmit power allocation algorithm is proposed to maximize the ratio of the learning accuracy of FL and the energy consumption of the IoT devices. The major contributions of this article are summarized as follows.

1) We propose to jointly optimize the selection of IoT devices and their transmit powers in an FL-empowered edge IoT system, thereby balancing the learning accuracy of FL and energy consumption of the IoT devices. The optimization is online, adapting to the time-varying arrivals of training data and the energy budget of the IoT devices, and the changing channel conditions between the IoT devices and the base station (i.e., the model aggregator).
2) FL-DLT3 is proposed to learn the network state dynamics while maximizing the ratio of the learning accuracy of FL to the energy consumption of the IoT devices. Considering the continuous transmit powers of the IoT devices, FL-DLT3 optimizes the edge server's selection of IoT devices for each round of FL and the transmit powers of the IoT devices, based on twin delayed deep deterministic policy gradient (TD3).
3) A new long short-term memory (LSTM) layer is designed in coupling with the proposed FL-DLT3 to predict the time-varying network states, e.g., data size, bandwidth, channel gain, and the remaining energy of the IoT devices. The LSTM layer estimates the unobserved states at every training iteration of the TD3. To the best of our knowledge, this is the first time that LSTM is employed in coupling with TD3 for the resource allocation of FL-enabled EdgeIoT.
4) FL-DLT3 is implemented in PyTorch. The effectiveness of FL-DLT3 is validated with the experimental data. Numerical results show that FL-DLT3 achieves fast convergence (less than 100 iterations) while the FL-accuracy-to-energy-consumption ratio is improved by 51.8%, as compared to the state of the art.

The remainder of this article is structured as follows. The literature on FL-based resource allocation in MEC is reviewed in Section II. Section III presents the FL protocol and system models. The resource allocation optimization for EdgeIoT is formulated in Section IV. Section V proposes the FL-DLT3 framework which conducts DRL-based EdgeIoT devices selection and resource allocation. Section VI evaluates the proposed FL-DLT3 framework. Finally, Section VII concludes this article.

II. Related Work

This section presents the literature on resource allocation with FL in MEC.

Assuming that the IoT devices have the same computational resources and wireless channel conditions, FedAvg [12] randomly selects IoT devices to participate FL training and synchronously aggregates local models. This process is repeated until a desirable training accuracy is achieved. In [18], different IoT devices own different computing capabilities and wireless channel conditions, resulting in different local model training time and upload time. The authors develop an FL protocol called FedCS, which allows the server to aggregate as many local model updates as possible to improve image classification accuracy.

To improve the training accuracy of FL systems in the context of wireless channels and energy arrivals of mobile devices, Chu *et al.* [19] modeled the transmission power allocation and mobile device selection of the FL training as a constrained Markov decision process (MDP). Due to a high complexity, stochastic learning methods and Lagrange multipliers are used to simplify the model and to obtain an efficient policy for all mobile device. In [20], the devices with limited battery energy, CPU computations, and bandwidths are considered in a mobile crowd network. A deep Q-learning-based resource allocation for data, energy, and CPU cycles is developed to reduce the energy consumption of FL-based mobile devices and training time of the FL. Given limited computation and communication resources at the devices, [21] analyzes the convergence bound of distributed gradient descent. A control algorithm is developed to determine the tradeoff between local update and global parameter aggregation. Some IoT devices have limited communication and computing resources and fail to complete training tasks, which leads to many discarded learning rounds affecting the model accuracy [22]. The authors study the multicriteria-based approach for IoT device selection in FL that reducing the number of communication rounds to reach the intended accuracy and increasing the number of selected IoT devices in each round.

Yoshida *et al.* [23] studied a trial-and-error-based IoT device selection based on multiarmed bandit (MAB), where computation tasks, traffic, and link qualities are unknown. The MAB-based IoT device selection balances the IoT device selection according to the selection frequency and IoT devices' resources. In [24], the IoT device scheduling in MEC is formulated without the IoT devices' channel and computing information. To reduce the training latency, the IoT device scheduling is formulated as an MAB problem, where ϵ-greedy is used to reduce the learning accuracy. Xia *et al.* [25] developed an MAB-based IoT device scheduling framework to reduce the training latency of FL. Given the known independent and identically distributed (i.i.d.) local data at the IoT devices, an FL-based IoT device scheduling algorithm is designed.

Zhu *et al.* [26] aimed to reduce the average age of data sources by controlling the IoT devices, scheduling the data, and allocating the bandwidth. An actor–critic learning framework is developed, where the IoT devices learn the scheduling strategies based on their local observations. To reduce the training time and energy consumption of IoT devices, Zhan *et al.* [27] designed an experience-driven algorithm based on proximal policy optimization and produce suboptimal results. To improve the network throughput, Kwon *et al.* [28] presented a cell association and base station allocation method. Multi-agent deep deterministic policy gradient is used to handle unexpected events and unreliable channels in underwater wireless networks. In [29], the IoT devices are selected to improve the FL accuracy and reduce the training time and energy consumption. To balance the training accuracy and delay of FL and energy consumption, twin delayed deep deterministic policy gradient algorithm (TD3) is employed to capture the interplay between function approximation error in both policy and value updates, and produce the IoT devices scheduling policy, the CPU frequency allocated for training, and the transmit power allocation.

An FL-based device selection optimization is developed in our preliminary work [30] to balance the energy consumption of the IoT devices and the learning accuracy of FL. The optimization model takes advantage of the *a-priori* knowledge of the network state information, e.g., data size, bandwidth, and channel gain. Due to the NP-hardness of the optimization, an energy efficiency-FL accuracy balancing heuristic algorithm (FedAECS) was presented in [30] to approximate the optimal IoT device selection policy offline. In contrast, this article considers a practical scenario without the prior information about data size, bandwidth, channel gain, and the remaining energy of the IoT devices. Due to a large state and action space, we propose a new FL-DLT3 to balance FL accuracy and energy consumption of the IoT devices online, where LSTM is leveraged to predict the hidden state of the IoT devices as the input state of TD3. In addition, we also compare the performance of the proposed FL-DLT3 with the FedAECS in [30].

III. System Model

In this section, we study the training protocol and energy model of FL. The notations used in this article are summarized in Table I.

TABLE I
List of Key Variables Defined in System Model

Notation	Definition
K	The total number of IoT devices
k	Index of IoT device
T	The total number of the rounds
t	Index of the rounds
$D_{t,k}$	IoT device k owns data size
$\mathbf{x}_{ki}$	Input of the FL model
y_{ki}	Output of the FL model
$\mathbf{w}$	Weight parameter of FL training
$\beta_{t,k}$	Whether IoT device k is selected
f_k	CPU frequency of IoT device k
ζ_k	Effective capacitance coefficient
L	Number of local iterations of FL training
$E_{t,k1}^{cmp}$	Energy consumption for computing $c_k D_{t,k}$ CPU cycles
$E_{t,k}^{cmp}$	Total computation energy at IoT device k
$r_{t,k}^{up}$	Achievable uplink transmit rate
$b_{t.k}$	Bandwidth allocated to IoT device k by the server
$P_{t,k}$	IoT device k transmit power
$G_{t,k}$	Uplink channel gain
$r_{t,k}^{down}$	Achievable downlink transmit rate of IoT device k
$H_{t,k}$	Downlink channel gain
P_t^s	Transmit power of the edge server
$\tau_{t,k}^{down}$	Downloading time of the global model at IoT device k
$\mathfrak{S}_d$	Size of the global model
$\tau_{t,k}^{up}$	Transmission time of the local model at t
$\mathfrak{S}_u$	Size of the local model
$E_{t,k}^{up}$	Energy consumption of device k on the local model transmission
$E_{t,k}^{c}$	Total energy consumption of the selected device k
$E_{t,k}$	Remaining battery energy of device k
$\Delta E_{t,k}$	Amount of harvested energy
Δ_t	Data evenness of selected devices
$\tau_{t,k}$	Completion time of IoT device k in t-th round
μ_k	System parameter
ν	Constant value

A. FL Protocol With MEC

Fig. 2 depicts the FL protocol, where the global model at the edge server and the local models at the IoT devices are trained in T rounds. Each FL round is composed of the resource request, IoT device selection, global model aggregation, global model download, local model update, and upload. The edge server initializes the hyperparameters of the global model, e.g., learning rate, batch size, and the weights of the global model.

1) `Resource Request:` The IoT devices send to the edge server the information needed, i.e., data size, for the device selection.
2) `IoT Device Selection:` The edge server selects the IoT devices to upload the local model for the training of the global model, where the details will be presented in the next section.

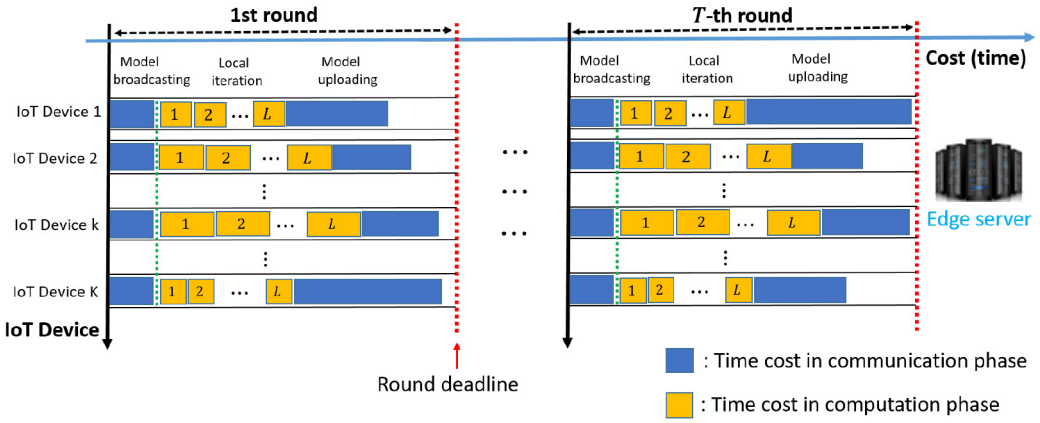


Fig. 2. Whole training process of FL, in each round, the time consumption of IoT device k includes global model download, local model update, and upload.

3) `Global Model Aggregation`: The edge server aggregates the local models of the selected IoT devices to produce the global model.
4) `Global Model Download`: All the IoT devices download the global model from the edge server.
5) `Local Model Update and Upload`: The selected IoT devices individually train their local models according to the FL parameters of the global model.

We consider K number of IoT devices, where $k \in [1, K]$. Let x_{ki} and y_{ki} denote the input (e.g., pixels of an image) and the output (e.g., labels of the image) of the FL model [31], respectively. The data set of device k is denoted as $\mathcal{D}_{t,k} = \{\mathbf{x}_{ki}, y_{ki}\}_{i=1}^{D_{t,k}}$, where $D_{t,k}$ is the size of the data set of device k in the tth round and data sample i in device k. Let $f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki})$ denote the loss function of FL, which captures approximation errors over the input $\mathbf{x}_{ki}$ and the output y_{ki}. $\mathbf{w}$ is the weight parameter of the loss function of the neural network being trained according to the FL procedure. Given $D_{t,k}$, the loss function at IoT device k can be specified as

$$F_{t,k}(\mathbf{w}) = \frac{1}{D_{t,k}} \sum_{i=1}^{D_{t,k}} f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki}) \tag{1}$$

where $f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki})$ can be specified according to the FL structure. For example, $f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki}) = (1/2)(\mathbf{x}_{ki}^T\mathbf{w} - y_{ki})^2$ is used to model linear regression, or $f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki}) = -\log(1 + \exp(y_{ki}\mathbf{x}_{ki}^T\mathbf{w}))$ is for the model of logistic regression [32]. Since the training of FL aims to minimize the weighted global loss function, we have

$$\begin{aligned} \min_{\mathbf{w}} F_t(\mathbf{w}) &= \sum_{k=1}^{K} \frac{D_{t,k}}{D_t} F_{t,k}(\mathbf{w}) \\ &= \frac{1}{D_t} \sum_{k=1}^{K} \sum_{i=1}^{D_{t,k}} f(\mathbf{w}, \mathbf{x}_{ki}, y_{ki}) \end{aligned} \tag{2}$$

where $D_t = \sum_{k=1}^{K} D_{t,k}$, it represents the total amount of data in the tth round.

B. Energy Model

The energy consumption of the IoT devices accounts for the local training of the data set and the transmissions of the local model. We assume that training a data sample at the IoT device requires c_k CPU cycles per bit. Given the data size of $D_{t,k}$, the number of CPU cycles for the local model training is $c_k D_{t,k}$. We denote f_k as the computation capacity of IoT device k, which is measured in CPU cycles per second. According to [33], the computation time of training the local model at device k in the each tth round [33], we have

$$\tau_{t,k}^{\text{train}} = \frac{c_k D_{t,k} L}{f_k} \tag{3}$$

where L is the number of local iterations of FL training.

According to [34] and [35], the energy consumption on $c_k D_{t,k}$ CPU cycles at IoT device k is

$$E_{t,k1}^{\text{cmp}} = \zeta_k c_k D_{t,k} f_k^2 \tag{4}$$

where ζ_k is the effective capacitance coefficient of computing chipset for device k. To compute the local model, IoT device k needs to compute $c_k D_{t,k}$ CPU cycles. Thus, the total computation energy at IoT device k in the tth round can be given as

$$E_{t,k}^{\text{cmp}} = L E_{t,k1}^{\text{cmp}} = L \zeta_k c_k D_{t,k} f_k^2. \tag{5}$$

Furthermore, the achievable uplink transmit rate of IoT device k is given by

$$r_{t,k}^{\text{up}} = b_{t,k} \log_2\left(1 + \frac{P_{t,k} G_{t,k}}{N_0 b_{t,k}}\right) \tag{6}$$

where $b_{t,k}$ is the bandwidth allocated to IoT device k by the edge server in the tth round, $P_{t,k}$ is the power consumption of data transmit for IoT device k, $G_{t,k}$ is the uplink channel gain between device k and the edge server, and N_0 is the power spectral density of the Gaussian noise.

The achievable downlink transmit rate of IoT device k is

$$r_{t,k}^{\text{down}} = b_{t,k} \log_2\left(1 + \frac{P_t^s H_{t,k}}{N_0 b_{t,k}}\right) \tag{7}$$

where P_t^s denotes the transmit power of the edge server, and $H_{t,k}$ is the downlink wireless channel gain from the edge server to device k.

The downloading time of the global model at device k in the tth round is

$$\tau_{t,k}^{\text{down}} = \frac{\mathfrak{S}_d}{r_{t,k}^{\text{down}}} \tag{8}$$

where $\mathfrak{S}_d$ denotes the size of the global model. The transmission time of the local model at the tth round can be given by

$$\tau_{t,k}^{\text{up}} = \frac{\mathfrak{S}_u}{r_{t,k}^{\text{up}}} \tag{9}$$

where $\mathfrak{S}_u$ is the size of the local model. By substituting (6) to (9), the energy consumption of an IoT device on the local model transmission is

$$E_{t,k}^{\text{up}} = P_{t,k}\tau_{t,k}^{\text{up}} = \frac{P_{t,k}\mathfrak{S}_u}{b_{t,k}\log_2\left(1+\frac{P_{t,k}G_{t,k}}{N_0 b_{t,k}}\right)}. \tag{10}$$

The total energy consumption $E_{t,k}^c$ of the selected IoT device k in the tth round is

$$E_{t,k}^c = E_{t,k}^{\text{cmp}} + E_{t,k}^{\text{up}}. \tag{11}$$

The remaining battery energy of the selected IoT device k in the tth round is

$$E_{t,k} = E_{t-1,k} - E_{t-1,k}^c + \Delta E_{t,k} \tag{12}$$

where $\Delta E_{t,k}$ is the amount of harvested energy.

IV. Problem Formulation

In this section, we study the IoT device selection and transmit power allocation to maximize the ratio of the learning accuracy of FL to the energy consumption of IoT devices.

Let $\beta_{t,k}$ be a binary indicator. If IoT device k is selected by the edge server in the tth round, $\beta_{t,k} = 1$; otherwise, $\beta_{t,k} = 0$. We define the accuracy of FL as the fraction of predictions FL model got right. According to [36]–[38], the accuracy of FL, denoted by $\Gamma(\beta_{t,k})$, can be simplified as

$$\Gamma(\beta_{t,k}) = \log\left(1+\sum_{k=1}^{K}\mu_k\beta_{t,k}D_{t,k}\right) \quad \forall t \in \mathcal{T} \tag{13}$$

where $\mu_k > 0$ is a system parameter [37]. To improve the evenness of data size of the selected IoT devices, similar to [39], we define the expectation of the difference between the total amount of data for all devices and the amount of data for the selected devices at the tth training round, and normalize the expectation, we have

$$\Delta_t = \frac{\mathbb{E}\left[\nu\sum_{k=1}^{K}D_{t,k} - \beta_{t,k}D_{t,k}\right]}{\sum_{k=1}^{K}D_{t,k}} \quad \forall t \in \mathcal{T} \tag{14}$$

where $\nu \in (0, 1]$ is a constant value, the arrival data $D_{t,k}$ of device k follows uniform distribution or normal distribution. The objective function can be determined as $([\Gamma(\beta_{t,k})]/[\sum_{k=1}^{K}\beta_{t,k}E_{t,k}^c]) - \Delta_t$, which aims to balance the learning accuracy of FL and energy consumption of the IoT device and improve the evenness of data size of the selected IoT devices. We formulate the optimization as **P1**

$$\mathbf{P1}: \max_{\beta_{t,k},P_{t,k}} \sum_{t=1}^{T}\left[\frac{\Gamma(\beta_{t,k})}{\sum_{k=1}^{K}\beta_{t,k}E_{t,k}^c} - \Delta_t\right]$$

$$\mathbf{s.t.}: \ \beta_{t,k}E_{t,k}^c \le E_{t,k}, \ (t \in [1,T], k \in [1,K]). \tag{15}$$

$$\Gamma(\beta_{t,k}) \ge \epsilon_0, \ (\epsilon_0 \in (0,1]). \tag{16}$$

$$\sum_{k=1}^{K}\beta_{t,k}b_{t,k} \le B, \ (t \in [1,T], k \in [1,K]). \tag{17}$$

$$1 \le \sum_{k=1}^{K}\beta_{t,k} \le K, \ (t \in [1,T], k \in [1,K]). \tag{18}$$

$$\beta_{t,k}\tau_{t,k} \le t_d, \ (t \in [1,T], k \in [1,K]). \tag{19}$$

$$P_k^{\min} \le P_{t,k} \le P_k^{\max}, \ (t \in [1,T], k \in [1,K]). \tag{20}$$

$$\beta_{t,k} \in \{0,1\}, \ (t \in [1,T], k \in [1,K]). \tag{21}$$

Specifically,

1) Constraint ($\beta_{t,k}E_{t,k}^c \le E_{t,k}$) guarantees that the selected IoT device has sufficient energy to complete the local model training in t.
2) Constraint ($\Gamma(\beta_{t,k}) \ge \epsilon_0$) specifies the minimum requirement of the FL accuracy in t, where $\epsilon_0 \in (0, 1]$ defines the lower bound threshold.
3) Constraint ($\sum_{k=1}^{K}\beta_{t,k}b_{t,k} \le B$) guarantees that the total bandwidth of the selected IoT devices is smaller than the bandwidth capacity B.
4) Constraint ($1 \le \sum_{k=1}^{K}\beta_{t,k} \le K$) describes that at least one IoT device is selected for FL.
5) Constraint ($\beta_{t,k}\tau_{t,k} \le t_d$) ensures that the download, computation and transmit delay of the selected device has to be less than the duration of the round t_d. As shown in Fig. 2, each tth round contains the downloading time of the global model, and local computation and transmission time of the local model. According to (3), (8), and (9), we can obtain the time delay $\tau_{t,k}$, i.e., $\tau_{t,k} = \tau_{t,k}^{\text{train}} + \tau_{t,k}^{\text{up}} + \tau_{t,k}^{\text{down}}$. Note that the IoT device selection time and training time of the global model at the edge server can be neglected since the edge server supports more powerful CPUs than the IoT device.
6) Constraint ($P_k^{\min} \le P_{t,k} \le P_k^{\max}$) indicates the upper and lower bounds of IoT devices' transmit power.

Problem **P1** involves nonlinear problems with continuous and integer variables [40]. The typical 0-1 multidimensional knapsack problem (MKP) [41] is a special case of problem **P1**. Assume that transmit power is a constant. In this case, the only variable is the selected IoT device. The items to be put in the knapsack are the IoT devices with energy consumption $E_{t,k}$, data size $D_{t,k}$, and bandwidth $b_{t,k}$. The capacity of the knapsack is equal to the total bandwidth, the optimization variable $\beta_{t,k}$ is a binary indicator of item (IoT device) k selection. $\beta_{t,k}$ is set to 1 to indicate that item k is selected. Otherwise, $\beta_{t,k}$ is set to 0. The total reliability of the knapsack has lower bounds which are equal to the minimum requirement of accuracy constraint (16). Note that every item to be put into the knapsack must obey the time constraint (19). Coupled with the energy consumption of IoT devices on the transmission is a nonlinear function of the transmit power, which has a continuous variable. Therefore, the proposed optimization is NP-hard. It is also mentioning that problem **P1** involves a long time horizon with random and unpredictable data and energy arrivals. This leads to an intractable large state space of the problem requires online optimization of selection and allocation decisions.

V. DRL-Based IoT Devices Selection and Resource Allocation

A. POMDP Formulation for FL-Based IoT Device Selection and Resource Allocation

The considered resource allocation can be formulated as a POMDP which is a generalization of an MDP with only partially observable states [42]. A POMDP can be represented by a 6-tuple (S, A, P, R, O, Ω), where S is the state space, A is the action space, P is the transition probability, R is the reward function, O is the observation space, and Ω is the observation model. The edge server cannot observe the underlying state. Instead, an observation $S^o_{\alpha'} \in O$ is received after a state transition to the next state $S_{\alpha'}$ with the probability $\Omega(S^o_{\alpha'}|S_{\alpha'})$.

State and Action Space: According to problem **P1**, the state space of the POMDP consists of data size, uplink channel gains, downlink channel gains, bandwidth, and the remaining energy of the IoT devices. The network state S_α is defined as

$$S_\alpha = \left\{\left(D_{\alpha,k}, G_{\alpha,k}, H_{\alpha,k}, b_{\alpha,k}, E_{\alpha,k}\right), k = 1, \ldots, K\right\}. \quad (22)$$

The action of the POMDP is the selection of the IoT devices for FL, denoted by $\beta_{\alpha,k}$, and the transmit powers of the selected IoT devices, $P_{\alpha,k}$, as given by

$$\mathcal{A} \in \left\{(\beta_{\alpha,k}, P_{\alpha,k}), k = 1, \ldots, K\right\} \quad (23)$$

where $\beta_{\alpha,k} \in \{0, 1\}$ and $P_{\alpha,k} \in [P^{\min}_k, P^{\max}_k]$.

Observation Space: At each state S_α, the edge server can observe partially the network state from the selected IoT devices, where the state observation S^o_α can be packed in its uploaded local model. Particularly, the state observation $S^o_\alpha \in S_\alpha$ is given by

$$S^o_\alpha = \left\{(D_{\alpha,k}, G_{\alpha,k}, H_{\alpha,k}, b_{\alpha,k}, E_{\alpha,k})_o, k = 1, \ldots, K\right\}. \quad (24)$$

Note that the state of unselected IoT devices cannot be observed by the edge server.

Reward: Let $R(S^o_{\alpha'}|S^o_\alpha, A_\alpha)$ denote the immediate reward received when the action $A_\alpha \in \mathcal{A}$ is taken at state S_α. The reward is defined to consist of the AE gain that is the ratio of the FL accuracy to the energy consumption of the selected IoT devices, and the penalty resulting from the data unevenness among the selected IoT devices, i.e.,

$$R\left(S^o_{\alpha'}|S^o_\alpha, A_\alpha\right) = \frac{\Gamma(\beta_{\alpha,k})}{\sum_{k=1}^{K} \beta_{\alpha,k} E^c_{\alpha,k}} - \Delta_\alpha \quad (25)$$

where $R(S^o_{\alpha'}|S^o_\alpha, A_\alpha)$ indicates that the state observation transits to subsequent $S^o_{\alpha'}$ from the current state observation S^o_α. For illustration convenience, we use R_α to denote the reward in the following sections.

To evaluate the action selected by a policy π_θ with parameters θ, where π_θ is a mapping from state observations to actions, and the set of all policies is defined as Π. We aim to maximize the expected total reward denoted as action-value function $Q_{\pi_\theta}(S^o_\alpha, A_\alpha)$

$$Q_{\pi_\theta}\left(S^o_\alpha, A_\alpha\right) = \max_{\pi \in \Pi} \mathbb{E}^{\pi_\theta}_{S^o_\alpha}\left\{\sum_{n=0}^{\infty} \gamma^n R_\alpha\right\} \quad (26)$$

where $\gamma \in [0, 1]$ is a discount factor for future state observations. $\mathbb{E}^{\pi_\theta}_{S^o_\alpha}\{\cdot\}$ takes the expectation with respect to policy π_θ and state observation S^o_α. According to the Bellman equation [43], the optimal action-value function (26) of a state–action pair (S^o_α, A_α) and the value of the subsequent state–action pair $(S^o_{\alpha'}, A_{\alpha'})$ can be further rewritten as

$$Q_{\pi_\theta}\left(S^o_\alpha, A_\alpha\right) = \max_{\pi_\theta \in \Pi} \mathbb{E}^{\pi_\theta}_{S^o_\alpha}\left\{R_\alpha + \gamma Q_{\pi_\theta}\left(S^o_{\alpha'}, A_{\alpha'}\right)\right\}. \quad (27)$$

The optimal action, namely, A^*_α, which satisfies (27), can be given by

$$A^*_\alpha = \arg\max_{\pi_\theta \in \Pi} \mathbb{E}^{\pi_\theta}_{S^o_\alpha}\left\{R_\alpha + \gamma Q_{\pi_\theta}\left(S^o_{\alpha'}, A_{\alpha'}\right)\right\} \quad (28)$$

where A^*_α provides the maximized AE gain.

Given a practical scenario where the edge server has no prior knowledge on the transition probabilities, we propose an FL-DLT3 framework that joint IoT device selection and transmit power allocation algorithm that utilizes TD3, one of the DRL techniques, to maximize the AE gain.

The state and action space increase dramatically with the number of devices, and the action space has both continuous and discrete actions. Moreover, the network state, in practice, is not always observable at the edge server because the data size and remaining energy information of IoT devices are always kept locally before the server takes the devices selection. In view of these challenges, it is difficult to find the exact solution of **P1**. To this end, we develop the FL-DLT3 framework to find the near-optimal solution of problem **P1**.

In general, FL-DLT3 consists of the TD3-based DRL and the LSTM-based state characterization layer, as depicted in Fig. 3. FL-DLT3 leverages the actor–critic neural network structure to develop the TD3-based DRL [44]. The TD3 at the edge server is trained to optimize IoT device selection and transmit power allocation in a continuous action space, where the edge server has no prior information on the state transition probabilities. The LSTM is used to predict the hidden state of the IoT devices as the input state of TD3. The AE gain is maximized over the large continuous state and action spaces. Specifically, the global model is trained iteratively to improve the learning accuracy of the local model. With the growth of FL iterations, FL-DLT3 optimally selects the IoT devices to maximize the accuracy of FL while ensuring the energy consumption requirement.

B. TD3 on the Edge Server

TD3 utilizes the deterministic policy gradient algorithm (DPG) [45] to optimally update the current policy by deterministically mapping network states to a specific action of the edge server. The critic is used to approximate the action-value function, which is related to the update of the actor, also known as policy. Moreover, the edge server stores the transition about the previous actions of the edge server, current observation, actions of the edge server, AE gain, next observations, i.e., $(A_{\alpha^-}, S^o_\alpha, A_\alpha, R_\alpha, S^o_{\alpha'})$, into the replay buffer $\mathcal{B}$ at each training step. A mini-batch of N transitions is randomly sampled from $\mathcal{B}$ to train the actor–critic networks. The action will be selected according to the policy network with exploration noise after the first M time steps. Since the IoT device selection action output from the policy network may not be

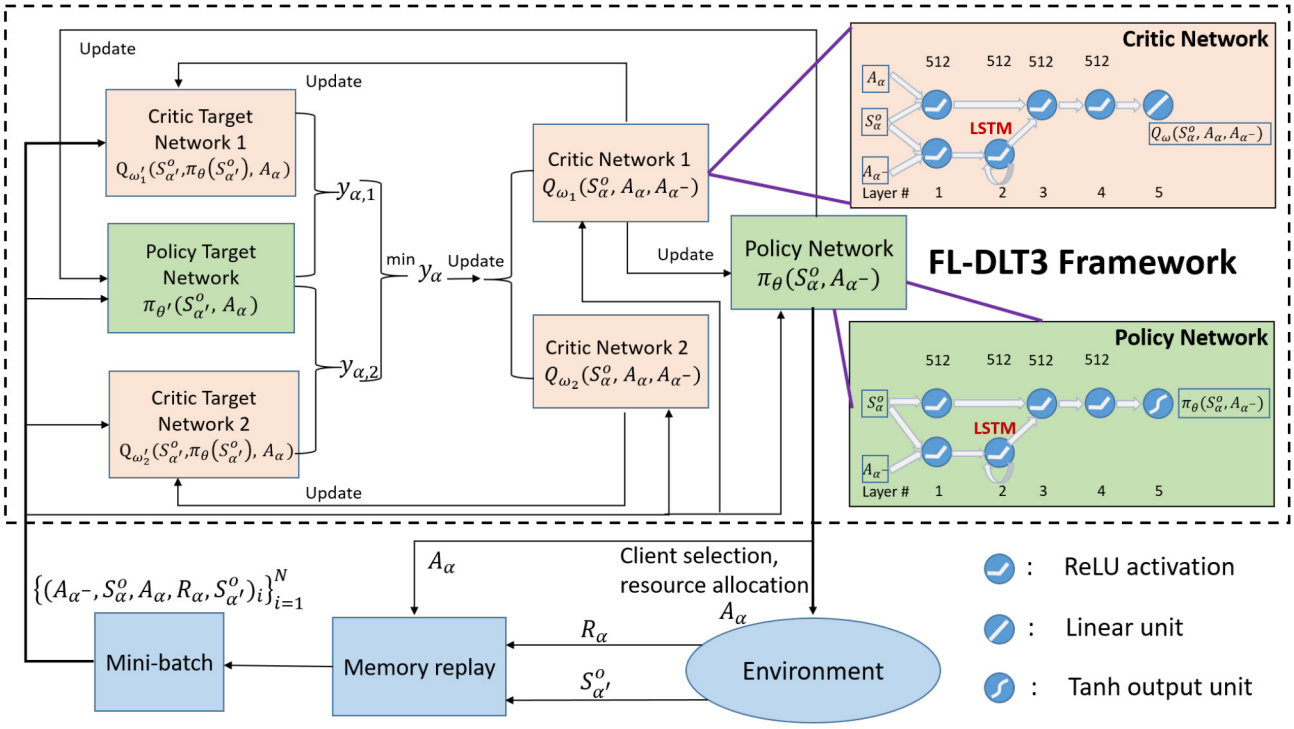


Fig. 3. Illustration of the proposed FL-DLT3 framework, where a policy network, a policy target network, two critic networks, and two critic target networks are trained. Each of the networks consists of a feedforward branch and a recurrent branch. A training round of FL is performed by the selected IoT devices with the allocated transmit power. Then, the edge server gets the reward R_α and the state update $S^o_{\alpha'}$ of the IoT devices. The transition $\{(A_{\alpha^-}, S^o_\alpha, A_\alpha, R_\alpha, S^o_{\alpha'})\}$ is stored into the replay buffer $\mathcal{B}$, and a mini-batch of transitions is randomly sampled from $\mathcal{B}$ to train the policy and critic networks on the edge server.

binary, we classify them as binary by setting thresholds. Based on the TD3 framework, the policy and critic networks can be trained to approximate the optimal policy of the formulated POMDP problem.

For the continuous IoT device selection and transmit power allocation, the objective is to find the optimal policy π_θ, which maximizes the expected AE gain $J(\theta) = \mathbb{E}^{\pi_\theta}_{S^o_\alpha}[R(S^o_0, A_0)]$, the parameterized policies π_θ can be updated by taking the gradient of the expected return with respect to θ, i.e.,

$$\nabla_\theta J(\theta) = \mathbb{E}_{\pi_\theta}\left[\nabla_{A_\alpha} Q_{\pi_\theta}(S^o_\alpha, A_\alpha)|_{A_\alpha=\pi(S^o_\alpha)} \nabla_\theta \pi_\theta(S^o_\alpha)\right] \quad (29)$$

where the optimal action-value function can be approximated by the critic neural network [46] $Q_\omega(S^o_\alpha, A_\alpha)$ with parameters ω, which obtains the AE gain. To update $Q_\omega(S^o_\alpha, A_\alpha)$, the critic neural network minimizes the approximation loss between the current target value and $Q_\omega(S^o_\alpha, A_\alpha)$ by adjusting the parameter ω

$$\min_\omega \mathbb{E}\left[\left(R_\alpha + \gamma Q_{\omega'}(S^o_{\alpha'}, \pi_\theta(S^o_{\alpha'})) - Q_\omega(S^o_\alpha, A_\alpha)\right)^2\right] \quad (30)$$

where ω' stands for the periodically updated parameters of the critic target network, $\pi_\theta(S^o_{\alpha'})$ is the policy target network takes action in the next state observation $S^o_{\alpha'}$.

However, the update of the critic neural network with the above value function may result in overestimation, which can produce suboptimal policies of the policy network. To deal with the overestimation bias problem, TD3 uses two approximately independent critic networks $\{Q_{\omega_1}, Q_{\omega_2}\}$ to estimate the value function, i.e.,

$$y_{\alpha,1} = R_\alpha + \gamma Q_{\omega_1'}(S^o_{\alpha'}, \pi_\theta(S^o_{\alpha'}))$$

$$y_{\alpha,2} = R_\alpha + \gamma Q_{\omega_2'}(S^o_{\alpha'}, \pi_\theta(S^o_{\alpha'})) \quad (31)$$

where the minimum of the two estimators is utilized in the update of the value function, i.e., $y_\alpha = \min\{y_{\alpha,1}, y_{\alpha,2}\}$.

Moreover, FL-DLT3 follows TD3 to update the policy and targets less frequently than the Q functions. Delayed policy updates consist of only updating the policy and critic target network every d intervals. And we also add noise to the target action, which makes the policy network less likely to exploit actions with high Q-value estimates. The proposed FL-DLT3 IoT device selection and transmit power allocation policy is shown in Algorithm 1. Since the network system has K IoT devices, each state of the IoT devices is made up of the source of data size, uplink channel gain, downlink channel gain, bandwidth, and the remaining energy of the IoT device. It takes T rounds of iterations to before the algorithm terminates. Therefore, the complexity of FL-DLT3 is $\mathcal{O}(T[7K(n_{pa}+n_{pc1}+n_{pc2}) + (l_{pa}-1)n^2_{pa} + (l_{pc1}-1)n^2_{pc1} + (l_{pc2}-1)n^2_{pc2} + 12 \times (7K \times N_{lstm} + (7K)^2 + 7K)])$, where n_{pa}, n_{pc1}, and n_{pc2} are the number of neurons in the hidden layer of policy network, critic network 1, and critic network 2, respectively; l_{pa}, l_{pc1}, and l_{pc2} are the number of the hidden layers of policy network, critic network 1, and critic network 2, respectively; and N_{lstm} is the hidden size of LSTM.

C. Network Architecture

Each network follows the two-branch structure as in [47], which consists of a feedforward branch and recurrent branch (as shown in Fig. 3). The feedforward branch and recurrent

Algorithm 1: Developed FL-DLT3 IoT Device Selection and Resource Allocation Policy

Initialize critic networks $\{Q_{\omega_1}(S^o_\alpha, A_\alpha, A_{\alpha^-}), Q_{\omega_2}(S^o_\alpha, A_\alpha, A_{\alpha^-})\}$ and policy network $\pi_\theta(S^o_\alpha, A_{\alpha^-})$ with random parameters $\omega_1, \omega_2, \theta$.
Initialize target networks $\omega_1^{'} \leftarrow \omega_1, \omega_2^{'} \leftarrow \omega_2, \theta' \leftarrow \theta$.
Initialize environment S^o_α, previous actions $A_{\alpha^-} \leftarrow 0$, replay buffer $\mathcal{B}$.
for $\alpha = 1, \cdots, T$ **do**
  if $\alpha \leq N$ **then**
    Explore N steps with random policy, obtain R_α and $S^o_{\alpha'}$, and storage the transition $(A_{\alpha^-}, S^o_\alpha, A_\alpha, R_\alpha, S^o_{\alpha'})$ of the N steps into $\mathcal{B}$.
  else
    Select action with exploration noise $A_\alpha \sim \pi_\theta(S^o_\alpha, A_{\alpha^-}) + \rho$, $\rho \sim \mathcal{N}(0, \sigma)$.
    Server allocates the selected IoT devices $P_{\alpha,k}$.
    Server performs model aggregation to obtain updated global model and distributes it to all selected devices in the next round.
    Server observes new observation $S^o_{\alpha'}$, calculates R_α, and stores the experience $(A_{\alpha^-}, S^o_\alpha, A_\alpha, R_\alpha, S^o_{\alpha'})$ into $\mathcal{B}$.
    Sample mini-batch of N experiences $\{(A_{\alpha^-}, S^o_\alpha, A_\alpha, R_\alpha, S^o_{\alpha'})i\}_{i=1}^N$ from $\mathcal{B}$.
    Obtain target action $\widehat{A}_{\alpha'} \leftarrow \pi_{\theta'}(S^o_{\alpha'}, A_\alpha) + \widehat{\rho}$, $\widehat{\rho} \sim \text{Clip}(\mathcal{N}(0, \widehat{\sigma}), -\text{c}, \text{c})$.
    $\widehat{y} \leftarrow R_\alpha + \gamma \min_{m=1,2} Q_{\omega_m}(S^o_{\alpha'}, \widehat{A}_{\alpha'}, A_\alpha)$.
    Update critics $\omega_m \leftarrow \arg\min_{\omega_m} \mathbb{E}(\widehat{y} - Q_{\omega_m}(S^o_\alpha, A_{\alpha^-}))^2$.
    if α *mod* $d = 0$ **then**
      Update θ by the deterministic policy gradient using equation (29).
      Update target networks:
      $\omega_{m'} \leftarrow \varphi\omega_m + (1-\varphi)\omega_{m'}$.
      $\theta' \leftarrow \varphi\theta + (1-\varphi)\theta'$.
    end
  end
end

branch both use five fully connected layers neural network of 512 hidden nodes. The recurrent branch consists of an embedding layer of 512 fully connected units followed by 512 LSTM units. For each hidden layer (apart from the LSTM), ReLU activations are used between the two hidden layers for both the policy and critic. The final tanh unit and linear unit following the output of the policy and critic, respectively.

D. LSTM Layer

The unknown network observation transitions resulting from time-varying data size, channel gains, bandwidth, and energy harvesting, which increase learning uncertainties and reduce learning accuracy. The edge server running the FL-DLT3 cannot observe the complete states of all the IoT devices. It can only make the observation of an IoT device when the device is selected and uploads its state information to the edge server. The learning efficiency and accuracy of the TD3-based FL-DLT3 can be compromised by the imperfect knowledge of the states of the IoT devices. Motivated by this fact, we develop a state characterization layer to predict the states of the IoT devices which are not observable, and feed the predicted states into every neural network of FL-DLT3.

For every policy (target) network, two critic (target) networks, and memory replay, we use LSTM to predict their respective hidden states. The LSTM-based neural network is used to find out the hidden state in the environment, e.g., the state of a device that has not been selected previously. Moreover, the LSTM-based state characterization layer helps to accelerate the convergence of the FL. At time state i, the hidden state h_i^{hid} is calculated by the following composite function:

$$h_i^{\text{hid}} = o_i \tanh(C_i) \tag{32}$$

$$o_i = \sigma\left(W_0 \cdot \left[C_i, h_{i-1}^{\text{hid}}, A_i\right] + e_0\right) \tag{33}$$

$$C_i = F_i C_{i-1} + p_i \tanh\left(W_c \cdot \left[h_{i-1}^{\text{hid}}, A_i\right] + e_c\right) \tag{34}$$

$$F_i = \sigma\left(W_f \cdot \left[h_{i-1}^{\text{hid}}, C_{i-1}, A_i\right] + e_f\right) \tag{35}$$

$$p_i = \sigma\left(W_p \cdot \left[h_{i-1}^{\text{hid}}, C_{i-1}, A_i\right] + e_p\right) \tag{36}$$

where o_i, C_i, F_i, and p_i denote the output gate, cell activation vectors, forget gate, and input gate of the LSTM layer, respectively. σ and tanh refer to the logistic sigmoid function and the hyperbolic tangent function, respectively. $\{W_0, W_c, W_f, W_p\}$ are the weight matrix, and $\{e_0, e_c, e_f, e_p\}$ are the bias matrix.

VI. Numerical Experiments

In this section, we present the implementation of the proposed FL-DLT3 on PyTorch, which is an open-source machine learning library based on the Torch library. We compare FL-DLT3 with benchmarks in terms of network size, data size, and communication bandwidth. Table II specifies the configuration of simulation parameters.

A. Implementation and Training of FL-DLT3

We implement the proposed FL-DLT3 with Python 3.9. PyTorch is set up on a Linux workstation with 64-bit Ubuntu 18.04. FL-DLT3 trains the resource allocation with FL on 2 Nvidia's GPUs, one is GeForce GTX 1060 with 3-GB memory, the other is GeForce RTX 2060 with 6-GB memory.

The experience replay memory can store 5×10^5 training samples in terms of previous actions of the edge server, current observation, actions of the edge server, AE gain, and next observations. The previous actions and current observation are combined to feed into policy network and critic networks to infer the hidden state. Both state space and action space are updated by using the Adam optimizer, where the learning rate is 3×10^{-4}. At one training step, the policy and critic networks are trained with a mini-batch of 45 transitions, sampled from the experience replay memory.

The policy target network is implemented by adding $\rho \sim \mathcal{N}(0, 0.5)$ to the actions chosen by the policy target network, clipped to (0, 60). The delayed policy updates the policy and critic target networks at every d interval, where $d = 10$.

B. AE Gain Performance

For performance validation, we compare FL-DLT3 with existing state-of-the-art FL-based device scheduling approaches, i.e., FedAECS [30], FedCS [18], and FedAvg [12].

TABLE II
SIMULATION PARAMETERS

Parameters	Values
Number of rounds (T)	1000
Number of local iterations (L)	4
For training one data sample CPU cycles per bit (c_k)	20 cycles / bit
Computation capacity of IoT device k (f_k)	[2, 4] GHZ
Transmit power of IoT device k ($P_{t,k}$)	[0.1, 60] W
Transmit power of the edge server (P_t^s)	[100,1000] W
Uplink channel gain ($G_{t,k}$)	$[10^{-3}, 10^{-1}]$ dB
Downlink channel gain ($H_{t,k}$)	$[10^{-1}, 10]$ dB
Power spectral density of the Gaussian noise (N_0)	1.0×10^{-8}
Parameter size of the global model (S_d)	1×10^4 bit
Upload data size of IoT device ($\mathfrak{S}_u$)	5×10^4 bit
Binary indicator of device selection ($\beta_{t,k}$)	{0, 1}
System parameter (μ_k)	4.2×10^{-9}
Effective capacitance coefficient (ϵ_k)	1.2×10^{-28}
The amount of harvested energy ($\Delta E_{t,k}$)	[50, 200] J
The constant value (ν)	1.0
Critic network learning rate	3×10^{-4}
Policy network learning rate	3×10^{-4}
Neural network weight coefficient (ϕ)	5×10^{-3}
Interval of policy target network update (d)	10
Discount factor (γ)	0.99
Batch size (N)	45
Replay buffer size ($\|\mathcal{B}\|$)	5×10^5
Exploration noise (σ)	0.5
Clipped normal noise (c)	0.5

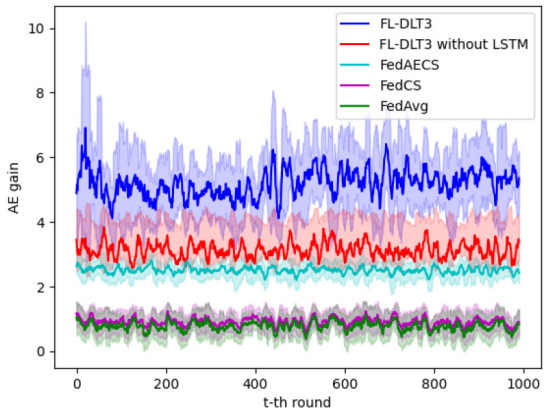


Fig. 4. Comparison of the AE gains, where $T = 1000$ and $K = 40$.

1) *FedAECS:* The edge server selects the IoT devices to fulfill a predetermined ratio of FL accuracy to energy consumption while meeting the requirement of accuracy and bandwidth.
2) *FedCS:* Given the bandwidth limit, the edge server selects the maximum number of IoT devices for FL.
3) *FedAvg:* Given the limit of the bandwidth, the edge server determines the number of IoT devices for FL training and randomly selects the IoT devices.

Fig. 4 shows the AE gains, where the tth round is from 1 to 1000 and $K = 40$. The data size $D_{t,k}$ and bandwidth $b_{t,k}$ of

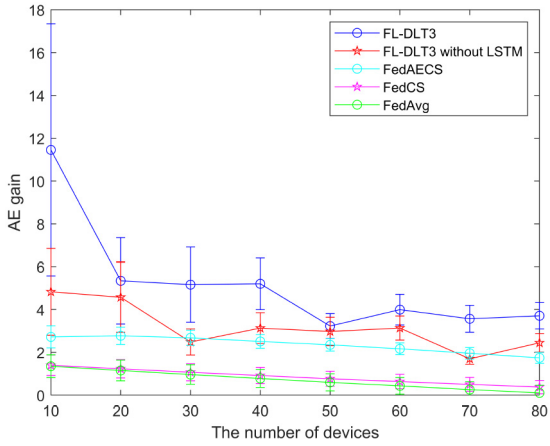


Fig. 5. Comparison of the AE gain obtained by using FL-DLT3, FL-DLT3 without LSTM, FedAECS, FedCs, and FedAvg with different number of IoT devices.

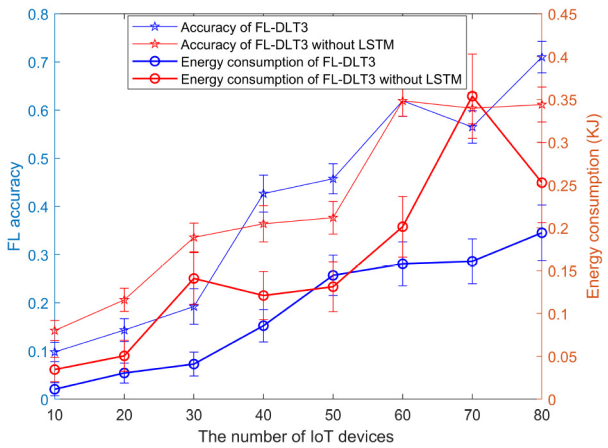


Fig. 6. Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the number of devices changes.

the K devices vary in [2, 10] MB and [10, 50] kHz, respectively. In general, the proposed FL-DLT3 achieves the highest AE gain, as compared to the existing FedAECS, FedCS, and FedAvg, improved by 51.8%, 82.4%, and 85.0% respectively. The reason is that FL-DLT3 leverages experience replay and predicts the states of the IoT devices which are not observable, however, FedAECS, FedCS, and FedAvg are unable to predict the states of the IoT devices.

FL-DLT3 outperforms FL-DLT3 without the LSTM layer with a gain of 39.8%. This is because the LSTM layer efficiently predicts the unknown network observation transitions, which enriches the training environment for FL and TD3.

Fig. 5 studies the AE gain of the proposed FL-DLT3, where K increases from 10 to 80. In general, the proposed FL-DLT3 achieves the highest AE gain 11.4557, as compared to the existing FedAECS (2.7212), FedCS (1.3952), and FedAvg (1.3498) given $K = 10$.

In Fig. 6, it can be observed that the energy consumption of FL-DLT3 dominates the performance when K increases from 10 to 50. When $K > 60$, the FL accuracy dominates the performance. This confirms that the AE gain's fluctuation in Fig. 5 when K increases from 10 to 50. The reason is that

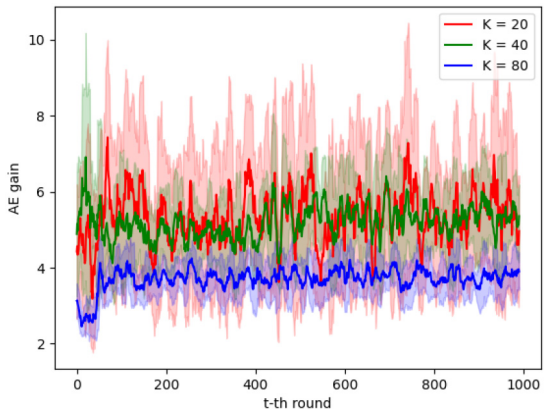


Fig. 7. AE gain achieved by the proposed FL-DLT3 given $K = 20$, 40, or 80.

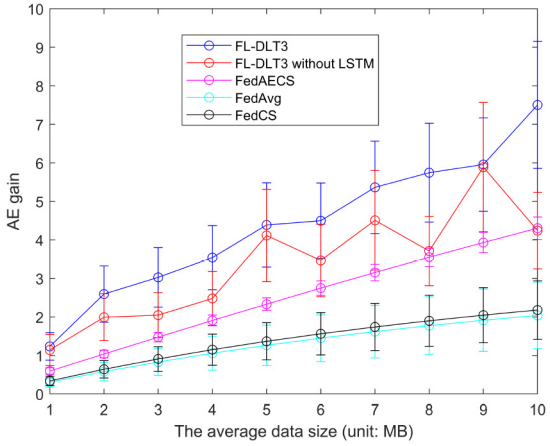


Fig. 8. AE gain varying with data size following normal distribution while keeping the same variance (0.2 MB).

FL-DLT3 can select more IoT devices, hence the FL accuracy and energy consumption increase monotonically with the number of devices.

Fig. 7 shows the AE gain given 1000 rounds. With an increase of K, the AE gain achieved by FL-DLT3 decreases from 5.3411 to 3.7066. This also validates the performance in Fig. 5.

Fig. 8 depicts the AE gain of FL-DLT3 with regard to the average data size. In general, the AE gain increases with the growth of the data size. This is because the FL accuracy rises in a high rate while the energy consumption of the IoT devices on the training of FL-DLT3 slightly increases. Moreover, the AE gain obtained by FL-DLT3 is about twice that obtained by FedAECS, thanks to the transmit power allocation in FL-DLT3.

Fig. 9 describes the AE gain of FL-DLT3 with regard to the average bandwidth. Overall, the AE gain raises with an increase of the average bandwidth since the energy consumption of the IoT devices on the local models transmission is reduced, in other words, a better channel quality leads to smaller packet retransmission and energy consumption. In addition, FL-DLT3 achieves the highest AE gain given different bandwidths. This is achieved by the LSTM layer is integrated into the FL-DLT3 to predict the time-varying bandwidth.

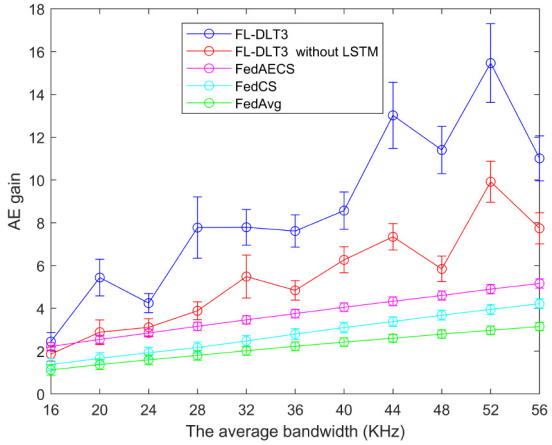


Fig. 9. AE gain varying with bandwidth following normal distribution while keeping the same variance (4 kHz).

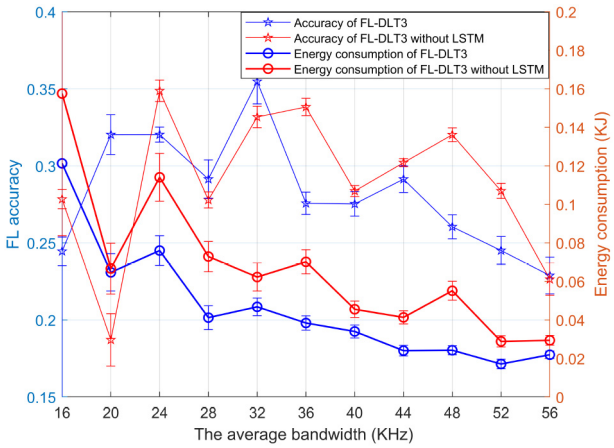


Fig. 10. Compare the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM as the average channel size of IoT devices changes.

Fig. 10 shows the FL accuracy and energy consumption of FL-DLT3 and FL-DLT3 without LSTM in regard to the average bandwidth size of IoT devices changes. Generally, the energy consumption of FL-DLT3 decreases with an increase of the bandwidth since the retransmission of the local models is reduced.

Fig. 11 shows the runtime of FL-DLT3, where K is set to 10, 40, 60, or 80. In the first 45 rounds, the runtime of FL-DLT3 is about 0.018 s. This is because the experience replay buffer is initialized in which the training of FL-DLT3 is not conducted yet. Once the learning experience is sufficient and FL-DLT3 carries out the training, the runtime raises to 0.74 s given $K = 10$. When K increases to 80, the training of FL-DLT3 takes 1.3286 s. The reason is that an increase of the IoT devices results in a large state and action space, hence the learning time of FL-DLT3 increases. In addition, the runtime of FL-DLT3 randomly fluctuates. This is due to random interruptions from other program executions that are concurrently operated on the server.

VII. Conclusion

In this article, we proposed FL-DLT3, which is a new DRL-based resource allocation with FL for EdgeIoT. Given

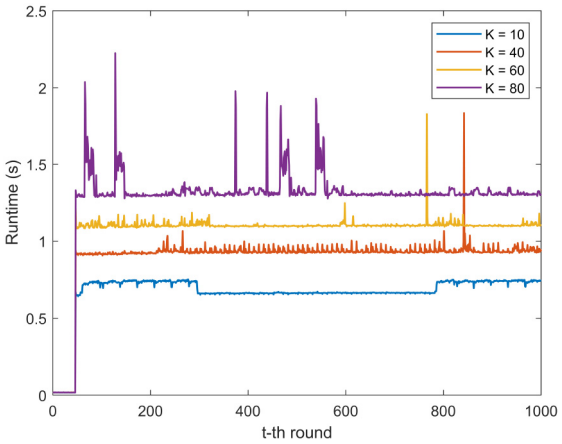


Fig. 11. Runtime with the tth round.

the large state and action space, FL-DLT3 learns the network state dynamics while maximizing the ratio of the learning accuracy of FL to the energy consumption of the IoT devices in a continuous domain. To improve the learning accuracy, a new state characterization layer based on LSTM is developed in FL-DLT3 to predict the time-varying data size, bandwidth, channel gain, and the remaining energy of the IoT devices. FL-DLT3 is implemented in PyTorch. The effectiveness of FL-DLT3 is validated with the experimental data.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] X. Sun and N. Ansari, "EdgeIoT: Mobile edge computing for the Internet of Things," *IEEE Commun. Mag.*, vol. 54, no. 12, pp. 22–29, Dec. 2016.

[3] S. Deng *et al.*, "Dynamical resource allocation in edge for trustable Internet-of-Things systems: A reinforcement learning method," *IEEE Trans. Ind. Informat.*, vol. 16, no. 9, pp. 6103–6113, Sep. 2020.

[4] P. Mach and Z. Becvar, "Mobile edge computing: A survey on architecture and computation offloading," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 3, pp. 1628–1656, 3rd Quart., 2017.

[5] O. A. Wahab, A. Mourad, H. Otrok, and T. Taleb, "Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems," *IEEE Commun. Surveys Tuts.*, vol. 23, no. 2, pp. 1342–1397, 2nd Quart., 2021.

[6] L. Xiao, X. Wan, C. Dai, X. Du, X. Chen, and M. Guizani, "Security in mobile edge caching with reinforcement learning," *IEEE Wireless Commun.*, vol. 25, no. 3, pp. 116–122, Jun. 2018. [Online]. Available: https://doi.org/10.1109/MWC.2018.1700291

[7] S. A. Rahman, H. Tout, C. Talhi, and A. Mourad, "Internet of Things intrusion detection: Centralized, on-device, or federated learning?" *IEEE Netw.*, vol. 34, no. 6, pp. 310–317, Nov./Dec. 2020.

[8] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. Artif. Intell. Stat.*, 2017, pp. 1273–1282.

[9] L. Xiao, X. Wan, X. Lu, Y. Zhang, and D. Wu, "IoT security techniques based on machine learning: How do IoT devices use AI to enhance security?" *IEEE Signal Process. Mag.*, vol. 35, no. 5, pp. 41–49, Sep. 2018.

[10] S. Ji, S. Pan, G. Long, X. Li, J. Jiang, and Z. Huang, "Learning private neural language modeling with attentive aggregation," in *Proc. Int. Joint Conf. Neural Netw. (IJCNN)*, 2019, pp. 1–8.

[11] Y. Li, X. Tao, X. Zhang, J. Liu, and J. Xu, "Privacy-preserved federated learning for autonomous driving," *IEEE Trans. Intell. Transp. Syst.*, early access, Jun. 16, 2021, doi: 10.1109/TITS.2021.3081560.

[12] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," 2016, *arXiv:1602.05629v1*.

[13] J. Kang, Z. Xiong, D. Niyato, S. Xie, and J. Zhang, "Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory," *IEEE Internet Things J.*, vol. 6, no. 6, pp. 10700–10714, Dec. 2019.

[14] W. Y. B. Lim, J. S. Ng, Z. Xiong, D. Niyato, C. Miao, and D. I. Kim, "Dynamic edge association and resource allocation in self-Organizing hierarchical federated learning networks," *IEEE J. Sel. Areas Commun.*, vol. 39, no. 12, pp. 3640–3653, Dec. 2021.

[15] J. Ji, X. Chen, Q. Wang, L. Yu, and P. Li, "Learning to learn gradient aggregation by gradient descent." in *Proc. IJCAI*, pp. 2614–2620.

[16] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, and M. Guizani, "A survey on federated learning: The journey from centralized to distributed on-site learning and beyond," *IEEE Internet Things J.*, vol. 8, no. 7, pp. 5476–5497, Apr. 2021.

[17] D. C. Verma *et al.*, "Federated learning for coalition operations," 2019, *arXiv:1910.06799*.

[18] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.

[19] S. Chu *et al.*, "Federated learning over wireless channels: Dynamic resource allocation and task scheduling," 2021, *arXiv:2106.06934*.

[20] T. T. Anh, N. C. Luong, D. Niyato, D. I. Kim, and L. Wang, "Efficient training management for mobile crowd-machine learning: A deep reinforcement learning approach," *IEEE Wireless Commun. Lett.*, vol. 8, no. 5, pp. 1345–1348, Oct. 2019. [Online]. Available: https://doi.org/10.1109/LWC.2019.2917133

[21] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 6, pp. 1205–1221, Jun. 2019.

[22] S. Abdulrahman, H. Tout, A. Mourad, and C. Talhi, "FedMCCS: Multicriteria client selection model for optimal IoT federated learning," *IEEE Internet Things J.*, vol. 8, no. 6, pp. 4723–4735, Mar. 2021.

[23] N. Yoshida, T. Nishio, M. Morikura, and K. Yamamoto, "MAB-based client selection for federated learning with uncertain resources in mobile networks," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, 2020, pp. 1–6.

[24] B. Xu, W. Xia, J. Zhang, T. Q. S. Quek, and H. Zhu, "Online client scheduling for fast federated learning," *IEEE Wireless Commun. Lett.*, vol. 10, no. 7, pp. 1434–1438, Jul. 2021.

[25] W. Xia, T. Q. S. Quek, K. Guo, W. Wen, H. H. Yang, and H. Zhu, "Multi-armed bandit-based client scheduling for federated learning," *IEEE Trans. Wireless Commun.*, vol. 19, no. 11, pp. 7108–7123, Nov. 2020.

[26] Z. Zhu, S. Wan, P. Fan, and K. B. Letaief, "Federated multiagent actor–critic learning for age sensitive mobile-edge computing," *IEEE Internet Things J.*, vol. 9, no. 2, pp. 1053–1067, Jan. 2022.

[27] Y. Zhan, P. Li, and S. Guo, "Experience-driven computational resource allocation of federated learning by deep reinforcement learning," in *Proc. IEEE Int. Parallel Distrib. Process. Symp. (IPDPS)*, 2020, pp. 234–243.

[28] D. Kwon, J. Jeon, S. Park, J. Kim, and S. Cho, "Multiagent DDPG-based deep learning for smart ocean federated learning IoT networks," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9895–9903, Oct. 2020.

[29] B. Yin, Z. Chen, and M. Tao, "Joint user scheduling and resource allocation for federated learning over wireless networks," in *Proc. IEEE Global Commun. Conf.*, 2020, pp. 1–6.

[30] J. Zheng, K. Li, E. Tovar, and M. Guizani, "Federated learning for energy-balanced client selection in mobile edge computing," in *Proc. Int. Wireless Commun. Mobile Comput. (IWCMC)*, 2021, pp. 1942–1947.

[31] A. Qayyum, K. Ahmad, M. A. Ahsan, A. I. Al-Fuqaha, and J. Qadir, "Collaborative federated learning for healthcare: Multi-modal COVID-19 diagnosis at the edge," 2021, *arXiv:2101.07511*.

[32] S. Shalev-Shwartz and S. Ben-David, *Understanding Machine Learning: From Theory to Algorithms*. Cambridge, U.K.: Cambridge Univ. Press, 2014.

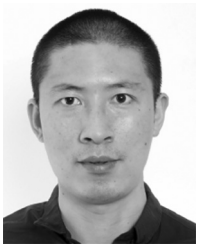
[33] C. T. Dinh *et al.*, "Federated learning over wireless networks: Convergence analysis and resource allocation," *IEEE/ACM Trans. Netw.*, vol. 29, no. 1, pp. 398–409, Feb. 2021. [Online]. Available: https://doi.org/10.1109/TNET.2020.3035770

[34] Z. Yang, M. Chen, W. Saad, C. S. Hong, and M. Shikh-Bahaei, "Energy efficient federated learning over wireless communication networks," *IEEE Trans. Wireless Commun.*, vol. 20, no. 3, pp. 1935–1949, Mar. 2021.

[35] Y. Mao, J. Zhang, and K. B. Letaief, "Dynamic computation offloading for mobile-edge computing with energy harvesting devices," *IEEE J. Sel. Areas Commun.*, vol. 34, no. 12, pp. 3590–3605, Dec. 2016.

[36] Y. Zhan, P. Li, Z. Qu, D. Zeng, and S. Guo, "A learning-based incentive mechanism for federated learning," *IEEE Internet Things J.*, vol. 7, no. 7, pp. 6360–6368, Jul. 2020.

[37] W. Y. B. Lim *et al.*, "Towards federated learning in UAV-enabled internet of vehicles: A multi-dimensional contract-matching approach," *IEEE Trans. Intell. Transp. Syst.*, vol. 22, no. 8, pp. 5140–5154, Aug. 2021.
[38] L. U. Khan *et al.*, "Federated learning for edge networks: Resource optimization and incentive mechanism," *IEEE Commun. Mag.*, vol. 58, no. 10, pp. 88–93, Oct. 2020.
[39] X. Lyu, C. Ren, W. Ni, H. Tian, R. P. Liu, and E. Dutkiewicz, "Optimal online data partitioning for Geo-distributed machine learning in edge of wireless networks," *IEEE J. Sel. Areas Commun.*, vol. 37, no. 10, pp. 2393–2406, Oct. 2019.
[40] N. V. Sahinidis, "Mixed-integer nonlinear programming 2018," *Optim. Eng.*, vol. 20, no. 2, pp. 301–306, 2019.
[41] H. Kellerer, U. Pferschy, and D. Pisinger, "Multidimensional knapsack problems," in *Knapsack Problems*. Berlin, Germany: Springer, 2004, pp. 235–283.
[42] L. P. Kaelbling, M. L. Littman, and A. R. Cassandra, "Planning and acting in partially observable stochastic domains," *Artif. Intell.*, vol. 101, nos. 1–2, pp. 99–134, 1998.
[43] R. Bellman, "Dynamic programming," *Science*, vol. 153, no. 3731, pp. 34–37, 1966.
[44] S. Fujimoto, H. Hoof, and D. Meger, "Addressing function approximation error in actor–critic methods," in *Proc. Int. Conf. Mach. Learn.*, 2018, pp. 1587–1596.
[45] D. Silver, G. Lever, N. Heess, T. Degris, D. Wierstra, and M. Riedmiller, "Deterministic policy gradient algorithms," in *Proc. Int. Conf. Mach. Learn.*, 2014, pp. 387–395.
[46] V. Mnih *et al.*, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[47] X. B. Peng, M. Andrychowicz, W. Zaremba, and P. Abbeel, "Sim-to-real transfer of robotic control with dynamics randomization," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, 2018, pp. 3803–3810.



Jingjing Zheng is currently pursuing the Ph.D. degree in electrical and computer engineering with the University of Porto, Porto, Portugal.

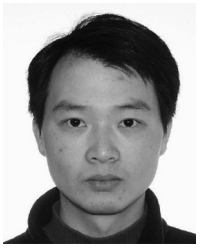
He is a Student Researcher with CISTER Research Center, Porto. His main research interests include federated learning, edge computing, and IoT security.

Kai Li (Senior Member, IEEE) received the B.E. degree from Shandong University, Jinan, China, in 2009, the M.S. degree from The Hong Kong University of Science and Technology, Hong Kong, in 2010, and the Ph.D. degree in computer science from The University of New South Wales, Sydney, NSW, Australia, in 2014.

He is currently a Senior Research Scientist with CISTER, Porto, Portugal. He is a CMU-Portugal Research Fellow, which is jointly supported by Carnegie Mellon University, Pittsburgh, PA, USA, and FCT, Lisbon, Portugal. Prior to this, he was a Postdoctoral Research Fellow with the SUTD-MIT International Design Center, The Singapore University of Technology and Design, Singapore, from 2014 to 2016. He was a Visiting Research Assistant with the ICT Center, CSIRO, Sydney, from 2012 to 2013. From 2010 to 2011, he was a Research Assistant with the Mobile Technologies Center, The Chinese University of Hong Kong, Hong Kong. His research interests include machine learning, vehicular communications and security, resource allocation optimization, cyber–physical systems, Internet of Things, and UAV networks.

Naram Mhaisen received the B.Sc. degree (with excellence) in computer engineering and the M.Sc. degree in computing from Qatar University (QU), Doha, Qatar, in 2017 and 2020, respectively. He is currently pursuing the Ph.D. degree with TU Delft, Delft, The Netherlands.

In 2020, he worked as a Research Assistant with the Department of Computer Science and Engineering, QU. His research interests include the design and optimization of (learning) systems with applications to networking.



Wei Ni (Senior Member, IEEE) received the B.E. and Ph.D. degrees in electronic engineering from Fudan University, Shanghai, China, in 2000 and 2005, respectively.

He is currently a Group Leader and a Principal Research Scientist with Commonwealth Scientific and Industrial Research Organization, Sydney, NSW, Australia, an Adjunct Professor with the University of Technology Sydney, Ultimo, NSW, Australia, and an Honorary Professor with Macquarie University, Sydney. Prior to this, he was a Postdoctoral Research Fellow with Shanghai Jiao Tong University, Shanghai, from 2005 to 2008, and a Deputy Project Manager with Bell Labs, Holmdel, NJ, USA, and Alcatel/Alcatel-Lucent, Boulogne-Billancourt, France, from 2005 to 2008. He was a Senior Researcher with the Devices Research and Development, Nokia, Espoo, Finland, from 2008 to 2009. His research interests include signal processing, stochastic optimization, as well as their applications to network efficiency and integrity.

Eduardo Tovar (Member, IEEE) was born in 1967. He received the Licentiate, M.Sc., and Ph.D. degrees in electrical and computer engineering from the University of Porto, Porto, Portugal, in 1990, 1995, and 1999, respectively.

He is currently a Professor with the Computer Engineering Department, School of Engineering (ISEP), Polytechnic Institute of Porto (IPP), Porto, where he is also engaged in research on real-time distributed systems, wireless sensor networks, multiprocessor systems, cyber–physical systems, and industrial communication systems. He heads the CISTER Research Unit, an internationally renowned research center focusing on RTD in real-time and embedded computing systems. Since 1991, he authored or coauthored more than 150 scientific and technical papers in the area of real-time and embedded computing systems, with emphasis on multiprocessor systems and distributed embedded systems.

Prof. Tovar is currently the Vice-Chair of ACM SIGBED (ACM Special Interest Group on Embedded Computing Systems) and was for five years, until December 2015, a member of the Executive Committee of the IEEE Technical Committee on Real-Time Systems.

Mohsen Guizani (Fellow, IEEE) received the B.S. (with Distinction), M.S., and Ph.D. degrees in electrical and computer engineering from Syracuse University, Syracuse, NY, USA, in 1985, 1987, and 1990, respectively.

He is currently a Professor of Machine Learning and the Associate Provost with Mohamed Bin Zayed University of Artificial Intelligence, Abu Dhabi, UAE. Previously, he worked in different institutions in the USA. He is the author of ten books and more than 800 publications. His research interests include applied machine learning and artificial intelligence, Internet of Things, intelligent systems, smart city, and cybersecurity.

Dr. Guizani has won several research awards, including the 2015 IEEE Communications Society Best Survey Paper Award, the Best ComSoc Journal Paper Award in 2021, as well five Best Paper Awards from ICC and Globecom Conferences. He was listed as a Clarivate Analytics Highly Cited Researcher in Computer Science in 2019–2021. He is also the recipient of the 2017 IEEE Communications Society Wireless Technical Committee Recognition Award, the 2018 AdHoc Technical Committee Recognition Award, and the 2019 IEEE Communications and Information Security Technical Recognition Award. He served as the Editor-in-Chief for IEEE NETWORK and is currently serving on the editorial boards of many IEEE transactions and magazines. He was the Chair of the IEEE Communications Society Wireless Technical Committee and the TAOS Technical Committee. He served as the IEEE Computer Society Distinguished Speaker and is currently the IEEE ComSoc Distinguished Lecturer.